Assessing Reviewers' Performance Based on Mining Problem Localization in Peer-Review Data

Wenting Xiong[1] and Diane Litman[1, 2] and Christian Schunn[2]
{wex12, dlitman, schunn}@pitt.edu
[1]Department of Computer Science, University of Pittsburgh
[2]Learning Research and Development Center, University of Pittsburgh

Abstract. Current peer-review software lacks intelligence for responding to students' reviewing performance. As an example of an additional intelligent assessment component to such software, we propose an evaluation system that generates assessment on reviewers' reviewing skills regarding the issue of problem localization. We take a data mining approach, using standard supervised machine learning to build classifiers based on attributes extracted from peer-review data via Natural Language Processing techniques. Our work successfully shows it is feasible to provide intelligent support for peer-review systems to assess students' reviewing performance fully automatically.

1 Introduction

Peers reviewing each other's writing is commonly used in various academic fields. A typical peer-review practice consists of three sections: first, students write essays on certain prompts; second, they provide feedback on essays of their peers; third, based on the feedback they get from their peers, they revise their initial essays. Peer-review is valuable not only because it provides learning opportunities for students, but also because it is more abundant in quantity compared with feedback from instructors. Besides, peer-review exercises also provide students the opportunity to develop their reviewing skills.

One problem with peer-review feedback is that its quality is not always guaranteed [1]. Previous studies on the nature of feedback suggest that the quality of feedback, in terms of the likelihood of its implementation, is significantly correlated with certain feedback features [2], among which problem localization is most significant. As defined in previous work, problem localization refers to pinpointing the source of the problem and/or solution. While such feedback features were used as mediators in the analysis of feedback helpfulness in [2], we believe that they could also be used as indicators in evaluating feedback quality automatically.

To date, current peer-review software facilitates the peer-review exercise with respect to document management and review assignment. However, no automatic feedback is generated for students regarding their reviewing performance. To add an automatic assessment component to peer-review software, we construct an evaluation system for reviewing performance that provides binary assessment for reviewers with respect to problem localization. Taking a data mining approach, we use standard supervised machine learning algorithms to build classifiers for identifying feedback features, based on attributes extracted from peer-review data with Natural Language Processing (NLP) techniques. Our results suggest that it is feasible to add an assessment component to peer-review software that could respond to students' reviewing performance automatically.

2 Related Work

Empirical studies of peer-review feedback based on manual coding have explored which feedback features predict whether feedback will be understood; the understanding of feedback was found to be a good predictor of whether feedback was implemented. For example, one study [2] has analyzed the rate of understanding the problem as a function of the presence/absence of feedback features, and found that feedback was more likely to be understood when the location of the problem was explicitly stated, or the solution to the specified problem was provided. This suggests those feedback features contribute to feedback implementation, which further indicates the helpfulness of feedback.

There is an increasing interest in research on computer-supported peer reviews that can bring benefits to both instructors and students. In our work, we aim to enhance the quality of feedback received by students by automatically assessing and guiding students' reviewing performance. Similarly, researchers from the data mining community have tried to predict feedback helpfulness automatically based on previous theoretical discoveries. With the help of software such as SWoRD[1], peer-review corpora are being collected and can be used for data mining and machine learning. One study [3] on a corpus that SWoRD collected used machine learning and classified any piece of peer-review feedback as helpful or not helpful based on tags that are automatically generated by tagging software. (In contrast, [2] took a manual-analysis approach: they require human annotators to code many feedback features that could be potentially relevant with respect to their purpose of study.) The result in [3] showed the performance of the classifier was limited by errors from the tagging software, which couldn't distinguish problem detection and solution suggestions (they are both types of criticism feedback).

In this paper we will also examine a corpus collected with SWoRD. However, in contrast to [3], we first detect the criticism feedback, and then predict the helpfulness of the recognized criticism feedback only based on the issue of problem localization. By treating criticism feedback as one group, we get around the problematic identification between solution suggestion and problem detection. In contrast to both [3] and our own prior work in [8], our system also aims to assess reviewing performance at the reviewer-level, rather than predicting the helpfulness [3] or problem localization [8] of a given piece of feedback. In the area of NLP, one related work of identifying criticism feedback could be sentiment analysis [4], while problem localization often involves paraphrasing [5] partial content of the associated essay. However, in this preliminary study, we take a simple approach in addressing these problems.

3 Data

The data used for this work is from a previous study [2] of the relationship between feedback features and helpfulness. The data was collected using SWoRD in a college

[1] Scaffolded Writing and Rewriting in the Discipline. http://www.lrdc.pitt.edu/Schunn/sword/index.html

level history introductory class. It consists of textual reviews provided by 76 reviewers referring to 24 associated student history essays.

In the previous study, all the textual reviews were manually segmented into 1405 idea-units (defined as contiguous feedback referring to a single topic).[2] These units were further annotated by two independent annotators for various coding categories. For the purpose of our work, we automatically predict two of the coding categories, *feedbackType* and *pLocalization*.

FeedbackType was coded with three values — criticism, praise and summary. For only criticism feedback, *pLocalization* was then coded as true or false, indicating whether the criticism feedback contains problem localization for any specified problems or suggested solutions. According to the coding scheme, *pLocalization* is not applicable to praise or summary feedback, thus its *pLocalization* was labeled as N/A. The reported agreement (Kappa) between two annotators on *FeedbackType* is 0.92, and that on *pLocalization* is 0.69. Relevant statistics are listed in Table 1. From now on, **feedback** will be used to refer to the 1405 annotated feedback idea-units.

Table 1. Descriptive statistics of annotations on history peer-review feedback data

Coding category	Value			
feedbackType	criticism	praise	summary	total
	875	388	142	1405
pLocalization	true	false	N/A	total
	462	413	530	1405

In addition to the feedback idea-units, we also have access to the collection of 24 essays to which the feedback refers. These essays provide domain knowledge, and are a self-contained resource that will assist us in mining features from the peer-review feedback data using statistical NLP techniques.

4 System and Features for Classification

Before diving into details of feature extraction and model learning, we would like to first provide an overview of our system, which takes the annotated feedback provided by a single reviewer, identifies target features sequentially for each piece of feedback, and generates assessment on the reviewer's reviewing performance with respect to problem localization in general (as the flow suggests in Figure 1).

[2] In the new version of SWoRD, segmentation is handled automatically through the interface which requires users to submit comments separately by idea-unit.

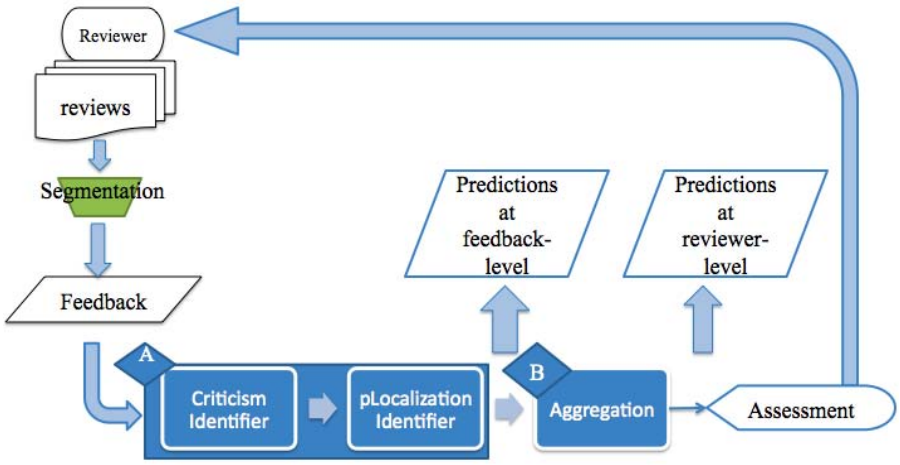

Figure 1. System overview

4.1 System overview

The system described in Figure 1 can be viewed as an assessment module compatible with peer-review software such as SWoRD. The system consists of two binary classifier components for identifying problem localization at the feedback-level (part A in Figure 1), plus an aggregation component (part B in Figure 1), which propagates the prediction of problem localization from the feedback-level up to the reviewer-level.

In pilot work, adding either annotated or predicted *feedbackType* into the feature set significantly improved the model's performance on identifying problem localization at the feedback-level. Therefore, we decompose the task into two concatenated tasks. We first use supervised learning to train a classifier for identifying criticism feedback versus praise and summary feedback; then we use the same algorithm to train another classifier for identifying whether problems are localized (*pLocalization* = true) for a given criticism feedback. Note that although both *feedbackType* and *pLocalization* were annotated with three values (Table 1), we combined values to create two binary classification tasks. Because it is the criticism feedback that is actionable, and focused on in the next step (classification for *pLocalization*), we group the praise and summary feedback together as non-criticism. As a byproduct, this binary separation also results in a more balanced data set from the perspective of machine learning. Similarly, recall that all non-criticism feedback in the data set was labeled N/A for *pLocalization*; we group N/A with false (vs. true), which simplifies our model for handling noisy output of the *feedbackType* identifier (specifically, any non-criticism predicted as criticism and sent to the second component).

Since our goal is to generate assessment of reviewing performance for each reviewer, we add another aggregation step into the system after the two components mentioned above, in which we make a binary decision on whether the reviewer provides enough problem localization in their reviews in general. This decision is based on the predictions made by the two preceding components on problem localization for all the feedback submitted by that reviewer. Since localization at the feedback-level is relatively difficult even for

humans (recall Kappa=0.69), we expect to provide more accurate (and hence useful) feedback at the aggregate level.

4.2 Criticism Identifier

To identify criticism feedback, we develop 3 groups of attributes that are automatically derived from the surface of sentences.

Simple: This set simply contains two attributes, the *wordCount* and the *feedbackOrder* in the review. WordCount is the number of words in the feedback; *feedbackOrder* is the index of the feedback with respect to its original review before segmentation. Based on our brief exploration of the data, we hypothesize that negative feedback is more likely to be verbose than positive feedback, and there is a certain pattern in expressing opinions, thus the *feedbackOrder* is useful in detecting criticism feedback.

Essay: There are four attributes in this group capturing the topic information contained in the feedback. To build a domain dictionary, first we preprocess the collection of 24 essays into bigrams (two adjacent words) and unigrams (single word). In particular, using NLTK[3] we extract bigrams whose term frequency–inverse document frequency (TF-IDF) is above the average TF-IDF of all bigram-collocations to form the bigram-domain dictionary. Then we gather all unigrams that constitute the bigrams in the bigram-domain dictionary for building our unigram-domain dictionary. Our final dictionary contains 291 bigrams and 402 unigrams. To capture how much content of the feedback is related to the domain, we count bigrams in the feedback that also belong to the bigram domain dictionary, and create the attribute *Collocation_d.* Similarly we create *Collouni_d* based on unigrams. Besides the domain-topics shared by all essays in general, we also considered essay-topics, referring to terms that are more frequently used in one specific essay rather than all of them. For each piece of feedback, we compute its bigrams and unigrams, and then only count (*Collocation_e* and *Collouni_e*) those that also appear in the associated essay with above-average item frequency of that essay. These four counts are normalized with the length of feedback.

Keyword: Due to the expensive computational cost for building models based on all words in the feedback corpus, we semi-automatically learned a set of Keywords (Table 2) which has categories based on the semantic and syntactic function of the words.[4] We first manually created a list of words that are specified as signal words for annotating *feedbackType* and *pLocalization* in the coding manual; then we supplemented the list

---

[3] Natural Language Toolkit. http://www.nltk.org/

[4] We also considered Bag-of-Words as well as sentiments that could be easily extracted with available software for opinion analysis features, but Keyword turned out to perform as well as the combination of these two groups, if not better. We prefer using Keyword in our model because it involves considerably fewer attributes thus reducing the complexity of our model, and it does not require the need for sentiment analysis software.

with the words selected by a decision tree model learned using a feature vector consisting of all words in the feedback (Bag-of-Words). As shown in Table 2, we generated nine attributes counting the number of words in the feedback that belongs to each tag category, respectively.

Table 2. Keyword table

Tag	Meaning	Word list
SUG	suggestion	should, must, might, could, need, needs, maybe, try, revision, want
LOC	location	page, paragraph, sentence
ERR	problem	error, mistakes, typo, problem, difficulties, conclusion
IDE	idea verb	consider, mention
LNK	transition	however, but
NEG	negative	fail, hard, difficult, bad, short, little, bit, poor, few, unclear, only, more, stronger, careful, sure, full
POS	positive	great, good, well, clearly, easily, effective, effectively, helpful, very
SUM	summarization	main, overall, also, how, job
NOT	negation	not, doesn't, don't

4.3 PLocalization Identifier

For a given piece of criticism feedback, we developed four groups of attributes to capture different perspectives of localized expressions.

Regular Expression: Three simple regular expressions were employed to recognize common phrases of location (e.g., "~~o~~n page 5", "~~th~~e section about"). If any regular expression is matched, the binary attribute *regTag* is true.

Domain Lexicon: Intuitively, localized feedback tends to use more domain vocabulary. Using the domain dictionary that we generated in 4.2 to calculate ESSAY attributes, we counted unigram domain-topics (*collouni_d*) contained in each piece of feedback.

Syntactic Features: Besides computing lexicon frequencies from the surface text, we also extracted information from the syntactic structure of the feedback sentences. We used MSTParser [6] to parse feedback sentences and hence generated the dependency structure of feedback sentences. Then we investigated whether there are any domain-topics between the subject and the object (*SO_domain*) in any sentence. We also counted demonstrative determiners (this, that, these and those) in the feedback (*DET_CNT*).

Overlapping-window Features: The three types of attributes above are based on our intuition about localized expressions, while the following attributes are derived from an overlapping-window algorithm that was shown to be effective in a similar task -- identifying quotations from reference works in primary materials for digital libraries [7]. To match a possible citation in a reference work, it searches for the most likely referred

window of words through all possible primary materials. We applied this algorithm for our purpose, and considered the length of the window (*windowSize*) plus the number of overlapped words in the window (*overlapNum*).

4.4 Example of Attribute Extraction

To illustrate how these attributes were extracted from the feedback, consider the following feedback as an example, which was coded as "*feedbackType*=criticism", "*pLocalization*=true":

> *The section of the essay on African Americans needs more careful attention to the timing and reasons for the federal governments decision to stop protecting African American civil and political rights.*

This feedback has 31 words (*wordCount* =31) and its index in the review is 2 (*feedbackOrder*=2). It has 2 bigram domain-topics ("African American" 2), 9 unigram domain-topics ("African" 2, "American", "Americans", "federal", "governments", "civil", "political" and "rights"), and 2 bigram essay-topics ("African American" 2) plus 8 unigram essay-topics (same as the unigram domain-topics except the "rights"). These four numbers are then normalized by the count of words in this feedback. As for Keyword, it contains 1 SUG ("need") and 2 NEG ("more", "careful").

The *regTag* is true because one regular expression is matched with "the section of"; there is no demonstrative determiner, thus *DET_CNT* is zero; "African Americans" is between the subject "section" and the object "attention", so *SO_domain* is true.

4.5 Aggregation

To finally generate an overall assessment of appropriate problem localization by each reviewer, the system aggregates the relevant predictions at the feedback level, calculating a *pLocalization%* score representing the reviewer's overall performance, which is the percentage of criticism feedback whose *pLocalization* is "true" submitted by that reviewer. To classify reviewers into "High" and "Low" groups regarding overall reviewing performance, we compare their *pLocalization%* score against a threshold and make a binary decision. In this work, we used an intuitive threshold that is the *pLocalization%* of criticism feedback of all reviewers (the number of "true" *pLocalization* criticism over the number of criticism feedback in the training data). This threshold performed best among several alternatives explored in a pilot study.

5 Experimental Setup

Though the system output is the assessment of reviewer's reviewing performance, its error could be due to any of the predictions made in the three components. To better analyze the predictive power of our system, we first evaluate each component separately (Section 5.1), then combine them and test its performance in a fully automatic version (Section 5.2). We compare our result to a Majority Class baseline for all experiments.

5.1 Component Evaluation

The *feedbackType* experiment uses all 1405 feedback from 76 reviewers. We use the Decision Tree algorithm provided by WEKA[5] and conduct 10-fold cross validation for evaluation. We chose the Decision Tree learning algorithm not only because it worked best in a preliminary study (compared with Naïve Bayes, Logistic Regression, and SVM), but also because the learned model (decision tree) is easier to interpret and provides clearer insights into how problem localization is recognized. Results are presented in Table 3 and explained in Section 6.

Recall that *pLocalization* was only coded for criticism feedback (non-criticism feedback are directly coded as N/A), thus for the isolated evaluation of this component, we only use the 875 criticism feedbacks for training and testing. The learning algorithm and evaluation settings are the same as those used for *feedbackType* (Table 3, Section 6).

5.2 System Evaluation

Though there is no annotation of overall problem localization for each reviewer (*pLocalization* at reviewer-level) in the data set, we can generate this by aggregating the *pLocalization* (annotated values) at the feedback-level, as we described in section 4.5. Note that when generating binary labels (High and Low) for each reviewer, the aggregation is based on annotated *pLocalization* and the threshold is calculated with annotated labels. When predicting, the aggregation is based on all predicted values. Thus the threshold would be different correspondingly.

When all components work together as a system, the *pLocalization* identifier receives the output of the *feedbackType* identifier. Therefore in the combined version, the *pLocalization* identifier was trained with the 1405 feedbacks, with one new attribute: predicted *feedbackType*. We again use the Decision Tree algorithm provided by WEKA for learning, while in this case we conduct leave-one-reviewer-out cross validation for evaluation. Results are presented in Table 4.

6 Results

Table 3 presents the experimental results of the performance of each component in isolation. With respect to the accuracy of our models, both significantly ($p<0.05$) outperform our baselines (79% vs. 62% and 78% vs. 53%) and their Kappa values are all greater than 0.5. Because we would like to provide further tutoring for the "Low" group of reviewers in the future, we are more interested in precision and recall of predicting the "Low" group. Thus we also analyze precision and recall for *feedbackType=criticism* and *pLocalization=true*, which are used to compute the *pLocalization%* scores. As listed in Table 3, both models achieve precision higher than 0.8, while for identifying criticism feedback the model's recall is even 0.86, though the *pLocalization* model has 0.73 for

[5] http://www.cs.waikato.ac.nz/ml/weka/

recall, which is still acceptable. Since the Majority class always predicts *feedbackType* as *criticism* and *pLocalization* as *true*, its recall will always be 1, thus we don't aim to beat the baseline for recall. Besides examining the quantitative results, we can also examine our results for qualitative characteristics. The learned model (decision tree) for *pLocalization* at the feedback-level is compact, using only 5 attributes (presented in [8]), and it suggests that domain-word counts plays an important role. The *feedbackType* model though more complicated, also relies on domain knowledge (Collocation_*d* and *Collouni_d* appear close to the root).

When all the components work together, the overall system can successfully predict the "Low" group of reviewers with both precision and recall higher than 0.8 (Table 4). Table 4 also presents the confusion matrix for details. Although system performance suffers from errors within each component, aggregation does alleviate it and maintains the overall performance comparable to the *pLocalization* model in isolation (M2 in Table 3).

Table 3. Performance of identification of feedbackType and pLocalization at feedback-level

	Model	Accuracy	Precision	Recall	Kappa
feedbackType (n = 1405)	Baseline	62%	0.62	1	0
	M1	79%	0.81	0.86	0.54
pLocalization (n=875)	Baseline	53%	0.53	1	0
	M2	78 %	0.82	0.73	0.55

Table 4. Performance of the overall system for identifying pLocalization at reviewer-level

Confusion matrix		
Predict\Label	High	Low
High	21	9
Low	8	38
Precision (Low)	0.81	
Recall (Low)	0.83	

7 Conclusion and Future Work

In this paper, we proposed a novel system for generating automatic assessments of reviewing performance with respect to problem localization at the reviewer-level. As a preliminary study in this new area of automatically assessing reviewing performance, we have demonstrated the feasibility of detecting reviewers who have low problem localization in reviewing, which is a first step for enhancing peer feedback quality. From the perspective of data mining, we successfully mine features of problem localization patterns from free form textual comments using statistical NLP techniques. Though we have started with simple methods and our classifiers are based on shallow NLP features, our system achieves comparatively high accuracy and precision for identifying reviewers who generally fail to provide localization information in their reviews.

In the future, we hope to construct a more complete dictionary of domain vocabulary, which might provide us with a better result based on our observations from this work. To improve the generalization of our system, we would also like to use a data driven approach to generate the Keyword list fully automatically (Table 2). Clearly each component in our system could be a NLP research topic, so we plan to explore the use of more sophisticated models from the NLP community as we discussed in the related work. Finally, since our ultimate goal is to help students with reviewing, we would like to perform a follow-up study to further evaluate how helpful the assessment generated by our system is in term of improving problem localization in future peer-review exercises for our "Low" reviewers.

References

[1] Kluger, A. N. & DeNisi, A. The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin*, 1996, 119(2), 254-284.

[2] Nelson, M.M. & Schunn, C.D. The nature of feedback: how different types of peer feedback affect writing performance. *Instructional Science*, 37, 2009, 375-401.

[3] Cho, K. Machine classification of peer comments in physics. *Educational Data Mining*, 2008.

[4] Wilson, T., Wiebe, J. M. & Hoffmann, Paul. Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, 2005, 347–354.

[5] Malakasiotis, P. Paraphrase recognition using machine learning to combine similarity measures. *Proceedings of the 47th Annual Meeting of ACL and the 4th Int. Joint Conf. on Natural Language Processing of AFNLP*, 2009.

[6] McDonald, R., Crammer, K. & Pereira, F. Online large-margin training of dependency parsers. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

[7] Ernst-Gerlach, A. & Crane, G. Identifying quotations in reference works and primary materials. *Research and Advanced Technology for Digital Libraries*, 2008, 78-87.

[8] Xiong, W. & Litman, D. Identifying problem localization in peer-review feedback. *Tenth International Conference on Intelligent Tutoring Systems*, 2010.